

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09237283 A**

(43) Date of publication of application: **09.09.97**

(51) Int. Cl. **G06F 17/50**

(21) Application number: **08043204**

(22) Date of filing: **29.02.96**

(71) Applicant: **RICOH CO LTD**

(72) Inventor: **YAMADA TAKAMITSU  
SUGAYA KAZUNOBU  
OKA ZENJI**

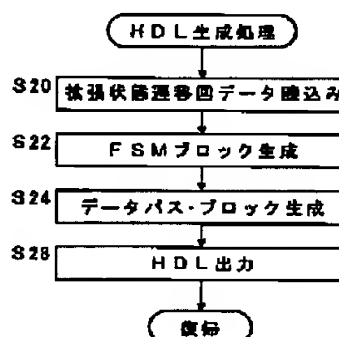
(54) **LSI FUNCTION DESIGN SUPPORT DEVICE**

(57) Abstract:

PROBLEM TO BE SOLVED: To generate hardware description data which corresponds to various targets and which can be logically synthesized with high quality from extension state transition drawing data.

SOLUTION: Hardware description data by HDL(hardware description language) is generated from extension state transition drawing data by permitting CPU to execute following operation based on a program stored in a memory. Description data by HDL of a part operating as a finite state machine is generated (S22) based on inputted extension state transition drawing data. Then, description data by HDL of a data path part is generated (S24). Data are outputted as hardware description data in a reregister transfer level being a design object (S28). Thus, hardware description data of LSI, in which an FSM part and the data path part are constituted as different blocks, can be obtained. Consequently, the logic synthesis tool of a high compression rate can be used on the FSM (finite state machine block) part and debugging after logic synthesis becomes easy.

COPYRIGHT: (C)1997,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-237283

(43) 公開日 平成9年(1997)9月9日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 17/50

識別記号

庁内整理番号

F I

G 0 6 F 15/60

技術表示箇所

6 5 4 A

審査請求 未請求 請求項の数7 O L (全 16 頁)

(21) 出願番号 特願平8-43204

(22) 出願日 平成8年(1996)2月29日

(71) 出願人 000006747

株式会社リコー

東京都大田区中馬込1丁目3番6号

(72) 発明者 山田 孝光

東京都大田区中馬込1丁目3番6号 株式会社リコー内

(72) 発明者 菅谷 和伸

東京都大田区中馬込1丁目3番6号 株式会社リコー内

(72) 発明者 岡 善治

東京都大田区中馬込1丁目3番6号 株式会社リコー内

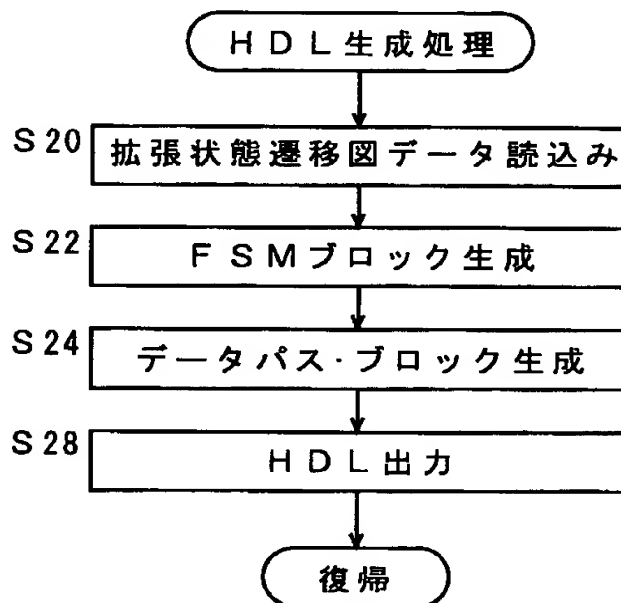
(74) 代理人 弁理士 青山 葆 (外1名)

(54) 【発明の名称】 L S I 機能設計支援装置

(57) 【要約】

【課題】 種々の目的に応じた質の高い論理合成が可能となるハードウェア記述データを拡張状態遷移図データから生成する。

【解決手段】 メモリに格納されたプログラムに基づきCPUが以下のように動作することにより、拡張状態遷移図データからHDLによるハードウェア記述データを生成する。入力された拡張状態遷移図データに基づき、まず有限状態機械として動作する部分のHDLによる記述データを生成し (S22)、次にデータパス部分のHDLによる記述データを生成し (S24)、これらのデータを設計対象のレジスタ転送レベルのハードウェア記述データとして出力する (S28)。これによりFSM部分とデータパス部分とが別のブロックとして構成されるLSIのハードウェア記述データが得られるため、FSM部分については圧縮率の高い論理合成ツールが使用でき、また論理合成後のデバッグ等が容易になる。



## 【特許請求の範囲】

【請求項1】 操作者による入力操作に基づき、任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データに基づき、前記LSIにおける制御論理部分に対応する有限状態機械の記述データを生成する有限状態機械記述生成手段と、前記拡張状態遷移図データに基づき、前記LSIにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするLSI機能設計支援装置。

【請求項2】 請求項1に記載のLSI機能設計支援装置において、前記データパス記述生成手段は、前記データパス部分における組合せ回路の記述データを生成する組合せ回路記述生成手段と、前記データパス部分におけるレジスタ群の記述データを生成するレジスタ記述生成手段とを有し、前記ハードウェア記述生成手段は、データパス部分が組合せ回路とレジスタ群とに分離された構成の前記LSIの前記ハードウェア記述データを生成する、ことを特徴とするLSI機能設計支援装置。

【請求項3】 前記制御論理部分を動作させるクロックと前記データパス部分を動作させるクロックとが異なる請求項1に記載のLSI機能設計支援装置において、前記LSIの外部からのテストモード信号およびテストクロック信号の入力が可能であって、該テストモード信号が所定値のときに、該テストクロック信号を、前記制御論理部分を動作させるクロック信号および前記データパス部分を動作させるクロック信号として供給するテスト回路の記述データを生成するテスト回路記述生成手段を備え、前記ハードウェア記述生成手段は、前記テスト回路記述生成手段によって生成された記述データを用いて、前記テスト回路が挿入された前記LSIの前記ハードウェア記述データを生成する、ことを特徴とするLSI機能設計支援装置。

【請求項4】 操作者による入力操作に基づき、状態遷移図により設計対象のLSIの動作を記述した状態遷移図データを生成する入力データ生成手段と、前記状態遷移図データによって表される前記LSIの各状態を識別する状態コードを生成する状態コード生成手段と、

前記LSIの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、マクロ選択手段によって選択されたマクロを前記状態コードの格納部とする前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするLSI機能設計支援装置。

【請求項5】 操作者による入力操作に基づき、状態遷移の階層的記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データによって表される前記LSIの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、前記LSIが前記マクロ状態にあるときに値が「1」となり、前記マクロ状態ではないときに値が「0」となるフラグ信号の記述データを生成するフラグ記述生成手段と、前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「1」のときに供給させ、前記フラグ信号が「0」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記遷移クロックの供給と停止が制御される前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするLSI機能設計支援装置。

【請求項6】 操作者による入力操作に基づき、状態遷移の階層的記述および任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データによって表される前記LSIの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、前記LSIが前記マクロ状態にあるときに値が「1」となり、前記マクロ状態ではないときに値が「0」となるフラグ信号の記述データを生成するフラグ記述生成手段と、前記マクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックを前記フラグ信号が「1」のときに供給させ、前記フラグ信号が「0」のときに該クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、

フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記クロックの供給と停止が制御される前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするLSI機能設計支援装置。

【請求項7】 操作者による入力操作に基づき、状態遷移の階層的記述および任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、  
前記拡張状態遷移図データによって表される前記LSIの各状態を識別する状態コードを生成する状態コード生成手段と、  
前記LSIの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、  
前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、  
前記拡張状態遷移図データによって表される前記LSIの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、  
前記LSIが前記マクロ状態にあるときに値が「1」となり、前記マクロ状態ではないときに値が「0」となるフラグ信号の記述データを生成するフラグ記述生成手段と、  
前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「1」のときに供給させ、前記フラグ信号が「0」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、  
前記拡張状態遷移図データおよびマクロ選択手段によって選択されたマクロに基づき、フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記LSIにおける制御論理部分である有限状態機械であってマクロ選択手段により選択されたマクロを前記状態コードの格納部とし前記フラグ信号により前記遷移クロックの供給と停止が制御される有限状態機械の記述データを生成する有限状態機械記述生成手段と、  
前記拡張状態遷移図データに基づき、前記LSIにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、  
有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴とするLSI機能設計支援装置。

## 【発明の詳細な説明】

### 【0001】

【発明の属する技術分野】 本発明は、LSI（大規模集積回路）の機能設計を支援する装置に関し、更に詳しくは、LSIの動作を状態遷移図形式で記述したデータからレジスタ転送レベルのハードウェア記述言語によるハードウェア記述データを生成する機能を備えたLSI機能設計支援装置に関する。

### 【0002】

10 【従来の技術】 近年、LSIの開発においてハードウェア記述言語(Hardware Description Language)（以下「HDL」という）を用いたトップダウン設計手法がとられるようになってきている。すなわち、機能設計の段階において設計対象のLSIをレジスタ転送レベルのHDLで記述したハードウェア記述データが作成され、このハードウェア記述データから論理合成ツールによってゲートレベルの回路データが生成されるようになってい

【0003】 このようなトップダウン設計手法において、レジスタ転送レベルのHDLによるハードウェア記述データの作成を容易にするために、設計者が状態遷移図を基本とする形式でグラフィック端末より設計対象のLSIの動作を示すデータ（設計情報）を入力するためのLSI機能設計支援装置が提案されている。この装置では、入力された状態遷移図形式のデータからレジスタ転送レベルのHDLによるハードウェア記述データが自動的に生成される。例えば、特開平3-41567号公報には、状態遷移図および機能図をCAD (Computer Aided Design) データとして対話形式でデータベース登録し、このデータベースより対応するハードウェア記述言語に変換してファイル出力する状態遷移図設計システムが開示されている。これによれば、デジタル回路の設計システムにおけるユーザインタフェイスが向上し、設計効率を向上させることができるという効果が得られる。

【0004】 上記のLSI機能設計支援装置によってハードウェア記述データを生成する場合、その記述スタイルによって、論理合成後の回路の品質や性能が異なる。このため、目的に応じた質の高い論理合成を行えるようなハードウェア記述が望まれる。

40 【0005】 例えば、論理合成後の回路規模の縮小化を重視する場合には、圧縮率の高い論理圧縮アルゴリズムによる論理合成ツールを使用できるようなハードウェア記述が望まれる。また、製造後の回路のデバッグやテストに手間を要すると予想される場合には、デバッグやテストが容易となるようなハードウェア記述が望まれる。ところで前述の状態遷移図によって表される各状態を示す状態コードを格納するフリップフロップは、通常、複数個から成り、同一クロックに同期する。したがって、クロックスキューが生じない安定した回路を実現するためには、状態コードを格納する複数のフリップフロップ

が一つの回路ブロックとして実現されるようなハードウェア記述が望まれる。また、近年の携帯機器の普及によりLSIの消費電力の低減が重要になっており、携帯機器に使用されるLSIについては、動作速度よりも消費電力を抑えることが優先され、これに対応したハードウェア記述が望まれる。

#### 【0006】

【発明が解決しようとする課題】しかし、従来のLSI機能設計支援装置では、ハードウェア記述データの生成において上記の点は考慮されていなかった。すなわち、設計者によって入力された状態遷移図形式のデータ（設計情報）により設計対象のLSIの動作が有限状態機械（以下「FSM」という）と所定状態におけるレジスタ動作の記述（データパスの記述）として表現されている場合には、このデータから従来のLSI設計支援装置によって生成されるハードウェア記述データに対しては、圧縮率の高いFSM専用の論理圧縮アルゴリズムに基づくツールを使用することができなかった。

【0007】また、従来のLSI機能設計支援装置には、状態遷移図形式の入力データからレジスタ転送レベルのハードウェア記述データを生成する際に、論理合成後の回路のデバッグやタイミング解析の容易化、消費電力の低減などについて考慮しているものはなかった。さらに、同一のクロックに同期する複数のフリップフロップについても、論理合成後にそれらが一つの回路ブロックとして実現されるようなハードウェア記述データを生成するLSI機能設計支援装置は存在しなかった。

【0008】そこで本発明では、回路規模（チップ面積）の縮小化や、デバッグ等の容易化、消費電力の低減、回路動作の安定化等の各種の目的に応じた質の高い論理合成が可能となるようなハードウェア記述データを状態遷移図形式の入力データから生成することができるLSI機能設計支援装置を提供することを目的とする。

#### 【0009】

【課題を解決するための手段】上記課題を解決するためになされた本発明に係る第1のLSI機能設計支援装置では、操作者による入力操作に基づき、任意状態におけるレジスタ動作の記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データに基づき、前記LSIにおける制御論理部分に対応する有限状態機械の記述データを生成する有限状態機械記述生成手段と、前記拡張状態遷移図データに基づき、前記LSIにおけるデータパス部分の記述データを生成するデータパス記述生成手段と、有限状態機械記述生成手段およびデータパス記述生成手段によって生成された記述データから、制御論理部分とデータパス部分とが分離された構成の前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備える

ことを特徴としている。

【0010】本発明に係る第2のLSI機能設計支援装置では、上記第1のLSI機能設計支援装置において、前記データパス記述生成手段は、前記データパス部分における組合せ回路の記述データを生成する組合せ回路記述生成手段と、前記データパス部分におけるレジスタ群の記述データを生成するレジスタ記述生成手段とを有し、前記ハードウェア記述生成手段は、データパス部分が組合せ回路とレジスタ群とに分離された構成の前記LSIの前記ハードウェア記述データを生成する、ことを特徴としている。

【0011】本発明に係る第3のLSI機能設計支援装置では、前記制御論理部分を動作させるクロックと前記データパス部分を動作させるクロックとが異なる上記第1のLSI機能設計支援装置において、前記LSIの外部からのテストモード信号およびテストクロック信号の入力が可能であって、該テストモード信号が所定値のときに、該テストクロック信号を、前記制御論理部分を動作させるクロック信号および前記データパス部分を動作させるクロック信号として供給するテスト回路の記述データを生成するテスト回路記述生成手段を備え、前記ハードウェア記述生成手段は、前記テスト回路記述生成手段によって生成された記述データを用いて、前記テスト回路が挿入された前記LSIの前記ハードウェア記述データを生成する、ことを特徴としている。

【0012】本発明に係る第4のLSI機能設計支援装置では、操作者による入力操作に基づき、状態遷移図により設計対象のLSIの動作を記述した状態遷移図データを生成する入力データ生成手段と、前記状態遷移図データによって表される前記LSIの各状態を識別する状態コードを生成する状態コード生成手段と、前記LSIの製造技術に対応する論理素子である複数ビット分のフリップフロップの機能を有するマクロが登録されているライブラリを記憶する記憶手段と、前記状態コードを表現するために必要なビット数分のフリップフロップの機能を有するマクロを前記ライブラリから選択するマクロ選択手段と、マクロ選択手段によって選択されたマクロを前記状態コードの格納部とする前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴としている。

【0013】本発明に係る第5のLSI機能設計支援装置では、操作者による入力操作に基づき、状態遷移の階層的記述が可能である拡張状態遷移図により設計対象のLSIの動作を記述した拡張状態遷移図データを生成する入力データ生成手段と、前記拡張状態遷移図データによって表される前記LSIの各状態が下位の状態を有する状態であるマクロ状態か否かを判定する状態判定手段と、前記LSIが前記マクロ状態にあるときに値が「1」となり、前記マクロ状態ではないときに値が

「0」となるフラグ信号の記述データを生成するフラグ記述生成手段と、前記マクロ状態における下位の状態の遷移クロックを前記フラグ信号が「1」のときに供給させ、前記フラグ信号が「0」のときに該遷移クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段と、フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記遷移クロックの供給と停止が制御される前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成するハードウェア記述生成手段と、を備えることを特徴としている。

【0014】本発明に係る第6のLSI機能設計支援装置では、上記第5のLSI機能設計支援装置において、上記ゲート記述生成手段に代えて、前記マクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックを前記フラグ信号が「1」のときに供給させ、前記フラグ信号が「0」のときに該クロックの供給を停止させるゲートの記述データを生成するゲート記述生成手段を備え、上記ハードウェア記述生成手段は、上記フラグ記述生成手段およびゲート記述生成手段によって生成された記述データを用いて、前記フラグ信号により前記クロックの供給と停止が制御される前記LSIをレジスタ転送レベルのハードウェア記述言語で記述したハードウェア記述データを生成することを特徴としている。

【0015】本発明に係る第7のLSI機能設計支援装置は、上記第1、第4、および第5のLSI機能設計支援装置における上記特徴を全て備えた構成としている。

【0016】

【発明の効果】本発明に係る第1のLSI機能設計支援装置によれば、制御論理部分（FSMの部分）とデータパス部分とが分離された構成のLSIのハードウェア記述データが生成されるため、FSMの部分の論理合成については、圧縮率の高いFSM専用の圧縮アルゴリズムによる論理合成ツールが使用可能となり、回路規模の縮小化を図ることができる。また、論理合成後の回路のデバッグやタイミング解析を、FSMとデータパス部分に分けて実施することができるため、デバッグやタイミング解析が容易となり、LSI設計の作業効率が向上する。

【0017】本発明に係る第2のLSI機能設計支援装置によれば、データパス部分が更に組合せ回路とレジスタ群とに分離された構成のLSIのハードウェア記述データが生成されるため、論理合成を組合せ回路部分とレジスタ群に分けて行うことができ、また、論理合成後の回路のデバッグやタイミング解析も組合せ回路部分とレジスタ群に分けて実施することができる。これにより、回路規模が更に縮小され、デバッグやタイミング解析が更に容易になる。

【0018】本発明に係る第3のLSI機能設計支援装

置によれば、LSI外部から入力されるテストモード信号によりテストクロック信号のFSMの部分およびデータパス部分への供給を可能とするテスト回路が挿入された構成のLSIのハードウェア記述データが生成される。これにより、機能設計の段階からテスト容易化設計が実施されることになり、論理合成後のテスト回路挿入のための作業が不要となるため、LSI設計の作業効率が更に向上する。

【0019】本発明に係る第4のLSI機能設計支援装置によれば、複数ビット分のフリップフロップの機能を有するマクロを状態コードの格納部とするLSIのハードウェア記述が生成され、これに対して論理合成を行うことにより、LSIにおいて状態コードを格納する部分が一つの回路ブロック（状態レジスタ）として実現される。これにより、クロックスキューの発生が防止され、LSIは安定に動作する。

【0020】本発明に係る第5のLSI機能設計支援装置によれば、生成されるハードウェア記述データに基づいて得られるLSIは、その状態がマクロ状態か否かを示すフラグ信号によりそのマクロ状態の下位の状態の遷移クロックの供給と停止が制御される構成となる。これにより、マクロ状態の下位の状態コードを格納する状態レジスタには、その下位の状態遷移が生じない期間中（そのマクロ状態以外の状態となる期間中）は遷移クロックの供給が停止されるため、LSIの消費電力が低減される。

【0021】本発明に係る第6のLSI機能設計支援装置によれば、生成されるハードウェア記述データに基づいて得られるLSIは、その状態がマクロ状態か否かを示すフラグ信号により、そのマクロ状態の下位の状態に割り付けられたレジスタ動作のためのクロックの供給と停止が制御される構成となる。これにより、マクロ状態の下位の状態に割り付けられたレジスタ動作に対しては、その下位の状態遷移が生じない期間中（そのマクロ状態以外の状態となる期間中）はクロックの供給が停止されるため、LSIの消費電力が低減される。

【0022】本発明に係る第7のLSI機能設計支援装置は、上記第1、第4、および第5のLSI機能設計支援装置の特徴を全て備えている。したがって、第7のLSI機能設計支援装置によれば、論理合成の効率化による回路規模の縮小、論理合成後の回路のデバッグやタイミング解析の容易化、状態レジスタにおけるクロックスキューの発生防止による動作の安定化、および消費電力の低減を図ることができる。

【0023】

【発明の実施の形態】

<実施形態1>以下、本発明の一実施形態（以下「実施形態1」という）であるLSI機能設計支援装置について説明する。このLSI機能設計支援装置は、汎用コンピュータとそれによって実行されるコンピュータ・プロ

グラムとによって実現される。

【0024】図1は、本実施形態のLSI機能設計支援装置のハードウェアであるコンピュータの構成を示す概略ブロック図である。本LSI機能設計支援装置のハードウェアは、CPU16とメモリ18などから成る本体10と、後述の拡張状態移行図データやライブラリデータなどを格納するハードディスク装置12と、キーボードやマウス等の入力装置14と、CRTディスプレイ等の表示装置20とから構成されている。このような構成において、本体10のメモリ18に所定のプログラムが格納され、CPU16がそのプログラムに基づいて動作することにより、LSIの機能設計を支援するための種々の機能が実現される。

【0025】図2は、上記LSI機能設計支援装置における処理を示すフローチャートである。上記LSI機能設計支援装置による設計では、まずステップS10において、設計者である操作者が、表示装置20を見ながらキーボードやマウス等の入力装置14を操作することにより、設計対象のLSIであるデジタル回路の動作を示す情報を入力する。本LSI機能設計支援装置では、状態移行図の形式で設計対象のLSIの動作を示す情報を入力することができ、このとき、その状態移行図で示される任意の状態における演算処理等のレジスタ動作を記述できるようになっている。具体的には、状態移行図で示される状態におけるデータパス部分の動作を示す代入文や演算式などをHDLで記述する。以下、このようなレジスタ動作の記述機能の追加等により拡張された状態移行図を「拡張状態移行図」という（後述の拡張状態移行図では、状態遷移の階層的記述の機能も追加されている）。このようなLSIの設計情報入力のための拡張状態移行図としては、例えば「オープンチャート」を用いることができる（山田孝光、安井隆、岡善治、「機能エントリ・ツールを用いたソーティング回路の設計」、CQ出版社、雑誌「インターフェース」、1995年3月号、p.160-166 参照）。

【0026】図4は、本LSI機能設計支援装置により入力された設計情報である拡張状態移行図の一例を設計対象のLSIの入出力の定義とともに示す図である。図4には、状態S1を示す円の横に「 $C=A+B$ 」と記載されており、これは、状態S1において実行される演算すなわちレジスタ動作を示している。この拡張状態移行図によって示される設計情報が操作者により入力されると、図5に示すようなデータ構造の設計データがメモリ18内に生成され、これがハードディスク装置12に拡張状態移行図データとして格納される。

【0027】このようにして設計情報が入力されて拡張状態移行図データが生成されると、ステップS12において、この拡張状態移行図データ対し所定のチェックを行う。すなわち、状態の遷移先が存在するか否か等の状態移行図に関するチェック（状態移行図チェック）や、

レジスタ動作を記述したハードウェア記述の文法チェック（HDL文法チェック）、論理合成の可能性のチェック（合成チェック）を行う。

【0028】上記各種のチェックの結果、エラーが発見されればステップS10へ戻り（ステップS14）、エラーがなくなるまでステップS10～S14が繰り返して実行される。そしてエラーがなくなればステップS16へ進み、操作者が所定の操作によりHDLの生成方式を指定する。すなわち、ハードウェア記述データの生成に際して、論理合成の効率（LSIのチップ面積の縮小化）を優先するか、消費電力の低減を優先するか、デバッグの容易性を優先するか等の優先事項を指定する。

【0029】HDL生成方式が指定されると、ステップS18において、拡張状態移行図データからハードウェア記述データを生成し（以下、このハードウェア記述データを生成する処理を「HDL生成処理」という）、LSI機能設計支援装置の処理を終了する。このようにして得られたハードウェア記述データは、設計対象のLSIをレジスタ転送レベルで記述したものであり、このハードウェア記述データから、既存の論理合成ツールにより、そのLSIの製造技術に対応するテクノロジーライブラリのマクロを用いて論理回路データが生成される。

【0030】図3は、図2のステップS18におけるHDL生成処理を実行するサブルーチンを示すフローチャートである。本LSI機能設計支援装置は、このフローチャートに示す処理により、上記の拡張状態移行図データからレジスタ転送レベルのハードウェア記述データを生成する。以下、このHDL生成処理について図3を参照しつつ説明する。なお本実施形態では、ハードウェア記述言語（HDL）としてVerilog-HDLを使用する。

【0031】HDL生成処理では、まずステップS20において、ハードディスク装置12に格納された拡張状態移行図データをメモリ18へ読み込む。次にステップS22において、読み込まれた拡張状態移行図データによって示されるFSMへの入力信号を入力ポートに持ち、FSMからの出力信号と現在の状態を示す信号（以下「状態信号」という）とを出力ポートに持つような一つの階層ブロックを設定し、これをFSMとして動作する階層ブロック（以下「有限状態機械ブロック」という）として記述したハードウェア記述データを生成する。例えば図4に示した拡張状態移行図データに対しては、有限状態機械ブロックのハードウェア記述データとして図8に示すようなデータを生成する。その後、ステップS24において、拡張状態移行図データによって示されるデータパス部分（図4に示した例では「 $C=A+B$ 」という記述に相当する部分）の入力バスと、前記FSMの階層ブロックの出力信号である状態信号とを入力ポートに持ち、データパス部分の出力バスを出力ポートに持つような一つの階層ブロックを設定し、これをデータパス部分として動作する階層ブロック（以下「データ

11

パス・ブロック」という)として記述したハードウェア記述データを生成する。例えば図4に示した拡張状態遷移図データに対しては、データパス・ブロックのハードウェア記述データとして図9に示すようなデータを生成する。このようにして、有限状態機械ブロックとデータパス・ブロックの両階層ブロックのハードウェア記述データを出力した後は、ステップS28において、図8に示す有限状態機械ブロックFSMの記述データおよび図9に示すデータパス・ブロックREGACTの記述データから成るハードウェア記述データを、これらの階層ブロックから構成される最上位の階層ブロックTOPのハードウェア記述データとして、すなわち図10に示すブロック図に相当するハードウェア記述データとして出力する。これによりHDL生成処理を終了し、このサブルーチンから復帰する。

【0032】以上のようにして本実施形態において生成されるハードウェア記述データと比較するために、従来の手法により図4に示した拡張状態遷移図データから生成されたハードウェア記述データを図6に示す。このときの設計対象のLSIの外部ポートの入出力関係は、図7に示すようになる。

【0033】本実施形態において図4に示した拡張状態遷移図データから生成されるハードウェア記述データは、前述のように図8および図9に示す通りであり、FSMに相当する部分とデータパスに相当する部分とが別個の階層ブロックとして構成されるようなハードウェア記述データとなっている。したがって、FSM専用の論理圧縮アルゴリズムによる論理合成ツールを、生成されたハードウェア記述のうちのFSMに相当する部分に適用することが可能となり、これにより高い圧縮率が得られる。また、論理合成後の回路デバッグやタイミング解析はFSMとデータパス部分を分離して実施することができるため、デバッグやタイミング解析が容易となり、設計の作業効率が向上する。

【0034】＜実施形態2＞上記実施形態において生成されるハードウェア記述データによれば、設計対象のLSIは有限状態機械ブロックとデータパス・ブロックとが分離された構成となるが、データパス・ブロックを、更に、組合せ回路のみから成る階層ブロックとレジスタ群からなる階層ブロックとに分離した構成となるようにすれば、論理合成やデバッグをこれらの階層ブロック毎に行うことができるため、論理合成の効率が更に向上し、論理合成後の回路のデバッグやタイミング解析が更に容易となる。このためには、図3のステップS24におけるデータパス・ブロックのハードウェア記述データの生成を以下のように変更すればよい(以下、このように変更した本発明の実施の形態を「実施形態2」という)。

【0035】すなわち、図11のフローチャートに示すように、まずステップS102において、ステップS2

(7)

12

0で読み込まれた拡張状態遷移図データによって示されるデータパス部分からクロック信号によって引き起こされるデータロード動作を削除したものを抽出し、これを一つの階層ブロック(以下「組合せ回路ブロック」という)として記述したハードウェア記述データを生成する。例えば図4に示した拡張状態遷移図データに対しては、組合せ回路ブロックのハードウェア記述データとして図12に示すようなデータを生成する。次にステップS104において、前記データパス部分のうち内部レジスタ(レジスタ群)を構成するフリップフロップへと論理合成によって変換されるものを抽出し、これを一つの階層ブロック(以下「レジスタ・ブロック」という)として記述したハードウェア記述データを生成する。例えば図4に示した拡張状態遷移図データに対しては、レジスタ・ブロックのハードウェア記述データとして図13に示すようなデータを生成する。このようにして得られた図12の組合せ回路ブロックCOMBの記述データと図13のレジスタ・ブロックREGの記述データから成るハードウェア記述データによれば、図14に示すように、データパス・ブロックREGACTにおいて組合せ回路ブロックCOMBとレジスタ・ブロックREGとが別個の階層ブロックとして構成されることになる。

【0036】＜実施形態3＞次に、テスト容易化設計に対応した本発明の実施の形態(以下「実施形態3」という)を説明する。本実施形態では、HDL生成処理は、図15のフローチャートに示すようになる。このフローチャートは、上記実施形態1における図3のフローチャートにテスト回路挿入のためのステップS26を追加したものである。その他の構成は、上記実施形態1と同様である。

【0037】本実施形態では、拡張状態遷移図データからレジスタ転送レベルのハードウェア記述データを生成する際に、上記実施形態1と同様に図15のステップS20～S24の処理を行った後、ステップS26においてテスト回路挿入のための処理を行う。すなわち、外部から供給されるテストモード信号とテストクロック信号に対するハードウェア記述データとともに、テストモード信号が「0」のときに有限状態機械ブロックとデータパス・ブロックとにそれぞれに対応した通常のクロックを供給し、テストモード信号が「1」のときにテストクロックを有限状態機械ブロックおよびデータパス・ブロックに供給するマルチプレクサに対するハードウェア記述データを生成する。これにより、図16に示す構成の回路に相当するレジスタ転送レベルのハードウェア記述データが得られる。図16は、最上位の階層ブロックTOPの構成を示すブロック図であって、この構成によれば、通常の動作時には、テストモード信号TEST\_MODEが「0」とされて、有限状態機械ブロックFSMにはクロック信号CLKが、データパスブロックREGACTにはクロック信号CK2がそれぞれ供給されるが、テスト時には、



テストモード信号TEST\_MODEが「1」とされて、有限状態機械ブロックFSMおよびデータパスブロックREGACTにテストクロック信号TEST\_CLOCKが供給される。

【0038】以上のように本実施形態によれば、レジスタ転送レベルのハードウェア記述データ生成においてテスト回路が挿入されるため、論理合成後においてテスト回路を挿入することなく、テスト容易化設計が実現される。これにより、LSI設計の作業効率が向上する。

【0039】＜実施形態4＞次に、本発明の第4の実施の形態（以下「実施形態4」という）であるLSI機能設計支援装置について説明する。本実施形態のLSI機能設計支援装置は、上記実施形態1～3と同一のハードウェア構成を有し（図1参照）、上記実施形態1～3と同様にして設計対象のLSIの動作を記述する拡張状態遷移図データを入力する（図2のステップS10～S14参照）。しかし本実施形態では、HDL生成処理が上記実施形態1～3と相違する。また本実施形態では、レジスタ転送レベルのハードウェア記述に使用できるマクロを集めたマクロ・ライブラリが、ハードディスク装置12に格納されている。そして、このマクロ・ライブラリには、設計対象のLSIの製造技術に対応するテクノロジー・ライブラリに登録されている複数ビット分のフリップフロップの機能を有するフリップフロップ・マクロに相当するマクロが登録されている。

【0040】以下、本実施形態のHDL生成処理について図17に示すフローチャートを参照しつつ説明する。まずステップS30において、実施形態1～3と同様にしてハードディスク装置12内に格納された設計情報である拡張状態遷移図データを、メモリ18に読み込む。以下では、このステップにおいて図18に示すような状態遷移図データが読み込まれたものとして説明する。図18において、例えば状態名「S0」はS0とラベル付けされた状態を示し、その横に次の状態として「S1」が記載されているのは、状態S0の次に状態S1に遷移することを示している。また、出力としての「Z」の記載の有無は、「Z」の記載された状態において信号名Zの出力信号が「1」となり、「Z」の記載されていない状態において信号名Zの出力信号が「0」となることを示す。なお図18に示した例では、レジスタ動作の記述は含まれていないため、メモリ18に読み込まれるデータは、通常の状態遷移図データである。

【0041】次のステップS32では、読み込まれた状態遷移図データによって表される状態を調べ、各状態に状態コードを割り付ける（状態コード解析処理）。図18に示した状態遷移図データの場合には、S0～S8の9個の状態が存在するため、最小ビット幅で状態コードを割り付けるとすると、4ビット幅の状態レジスタが必要となる。図19は、図18の状態遷移図データに対する状態コード解析の結果を示しており、この例では、状態S0～S8に4ビットの状態コード「0000」～

「1000」がそれぞれ割り付けられている。このようにして状態コードが割り付けられると、設計対象のLSIの動作を示す状態遷移図は図22に示すようになり、これが表示装置20に表示される。

【0042】状態コードの割り付け後は、ステップS34において、ハードディスク装置12に格納されたマクロ・ライブラリを参照して、状態コードを格納する状態レジスタの実現に必要なフリップフロップ・マクロを求める。いま、マクロ・ライブラリには、図20に示すように、「R42」というマクロ名の4ビットのフリップフロップ・マクロと、「R82」というマクロ名の8ビットのフリップフロップ・マクロとが登録されているものとする。この場合、図19に示したように状態コードが割り付けられたとすると、フリップフロップ・マクロR42が選択され、これに対応する図21に示すデータが、設計対象のLSIのハードウェア記述に使用されるフリップフロップ・データとしてハードディスク装置12に格納される。

【0043】そしてステップS36において、ステップS30で読み込まれた状態遷移図データから、ステップ34で得られたフリップフロップ・データを参照して、レジスタ転送レベルのハードウェア記述データを生成する。

【0044】以上のようにして本実施形態において生成されるハードウェア記述データと比較するために、従来のLSI機能設計支援装置により図18の状態遷移図データから生成されたハードウェア記述データを図23に示す。図23において、HDLの記述201は4ビットの状態コードの割り付けを、HDLの記述202は状態レジスタをそれぞれ示す。この図23に示すハードウェア記述データから論理合成ツールによって論理回路を合成すると、図25に示すような回路が得られる。

【0045】これに対し、本実施形態において図18の状態遷移図データから生成されるハードウェア記述データは、図24に示すようになる。図24において、HDLの記述211は状態コードの割り付けを、HDLの記述212は状態レジスタを、HDLの記述213は4ビットのフリップフロップ・マクロR42の使用をそれぞれ示している。この図24に示すハードウェア記述データから論理合成ツールによって論理回路を合成すると、図26に示すような回路が得られる。従来のLSI機能設計支援装置により得られた図25に示した論理回路では、状態コードを格納するレジスタとして1ビットのフリップフロップDFFCORが4個生成されているのに対し、図25に示す論理回路では、状態コードを格納するレジスタとして4ビットのフリップフロップ・マクロR42が1個生成されている。

【0046】このように本実施形態によれば、状態コードを格納する状態レジスタとして複数ビットのフリップフロップ・マクロを使用したハードウェア記述データが

得られる。このフリップフロップ・マクロは、設計対象のLSIの製造技術に対応するテクノロジー・ライブラリに登録されているマクロに相当するものである。このため、上記ハードウェア記述データに対して論理合成を行うと、状態レジスタとして複数ビット分のフリップフロップが一つの回路ブロックとして生成される。これにより、クロックスキュー等が生じず安定に動作するLSIを実現することができる。また、状態レジスタが一つのマクロとして実現されるため、チップ面積の点でも有利である。

【0047】＜実施形態5＞次に、本発明の第5の実施形態（以下「実施形態5」という）であるLSI機能設計支援装置について説明する。本実施形態のLSI機能設計支援装置も、上記実施形態1～4と同一のハードウェア構成を有し（図1参照）、上記実施形態1～4と同様にして設計対象のLSIの動作を記述する拡張状態遷移図データを入力する（図2のステップS10～S14参照）。ただし、本実施形態で使用される拡張状態遷移図は、任意の状態におけるレジスタ動作を記述できるのみならず、状態遷移を階層的に記述できるものである。例えば前述のオープンチャートは、このような記述機能を有している。

【0048】本実施形態のHDL生成処理は上記実施形態1～4と相違する。以下、本実施形態のHDL生成処理について図27に示すフローチャートを参照しつつ説明する。まずステップS40において、階層的記述が可能な拡張状態遷移図によって表現された設計情報である拡張状態遷移図データを、ハードディスク装置12からメモリ18に読み込む。図28は、ステップS40で読み込まれる拡張状態遷移図データの一例を示す図である。この拡張状態遷移図データによって表される状態のうち、状態HS0はその下階層として拡張状態遷移図304で示されるような状態遷移を持ち、状態HS1およびHS2も下階層を持っている。また、状態HS0の下階層における状態S0に対するHDLの記述303は、レジスタ動作を示すものであり、HDLの記述302は、このレジスタ動作に対して供給されるクロックを定義するものである。

【0049】上記ステップS40において拡張状態遷移図データが読み込まれた後は、これによって表される状態を順次調べながら、ハードウェア記述データを生成する。すなわち、まずステップS42において、未調査の状態の有無を判定し、未調査の状態があればステップS44へ進んで未調査の状態を一つ求める。このとき、同一階層に未調査の状態が存在するときは同一階層における未調査の状態を選択し、同一階層に未調整の状態が存在しないときはその上位の階層における未調査の状態を選択する。

【0050】次のステップS46では、ステップ44で求められた状態が下階層の状態遷移を持つか（下位の状

態を持つか）否かを判定し、下階層の状態遷移を持たない場合は、ステップS48へ進んで、ステップS44で求められた状態から出ていく矢印が有るか否かを判定する。その結果、矢印が無い場合はステップS42に戻り、矢印が有る場合はステップS50へ進む。

【0051】ステップS50では状態の遷移条件を求め、次のステップS52において、調査対象の状態（ステップS44で求められた状態）に対してレジスタ動作が記述されていれば、そのレジスタ動作を示すハードウェア記述データを生成する。その後、ステップS54において状態の遷移先を求める。そしてステップS56において、このようにして求められた遷移条件および遷移先（次の状態）と現在の状態とに基づき、FSMに対するハードウェア記述データを生成する。その後、ステップS42に戻る。以後、未調査の状態が存在しかつステップS44で求められる状態が下階層の状態遷移を持たない限り（ステップS46参照）、ステップS42→S44→S46→S48→S50→S52→S54→S56→S42というループ（ただし、調査対象の状態から出ていく矢印が無いときはステップS48からS42へ戻る）を繰り返し実行する。この実行中にステップS46において下階層の状態遷移を持つと判定されると（以下、下階層の状態遷移を持つ状態を「マクロ状態」という）、ステップS58へ進む。

【0052】ステップS58では、その下階層でのみ動作するクロックに対するハードウェア記述データを生成する。具体的には、まず、その下階層の上位の階層の状態遷移図における状態コードがこの状態（その下階層の状態遷移を持つマクロ状態）であることを示す値のときに「1」となり、それ以外のように「0」となるようなフラグ信号のハードウェア記述データを生成し、次に、このフラグ信号と下階層の状態遷移用として供給すべき元のクロックとの論理積の信号を、下階層の状態コードを格納する状態レジスタへクロックとして供給することを示すハードウェア記述データ、すなわち上記フラグ信号を用いたANDゲートによるクロックのゲーティングを示すハードウェア記述データを生成する。

【0053】上記ステップS58の実行後は下階層の状態遷移に移行し、未調査の状態が存在しかつステップS44で求められる状態が下階層の状態遷移を持たない限り（ステップS46参照）、S42→S44→S46→S48→S50→S52→S54→S56→S42というループ（ただし、調査対象の状態から出ていく矢印が無いときはステップS48からS42へ戻る）を繰り返し実行する。これにより、下階層の状態遷移図について上記と同様の処理が行われる。この実行中にステップS46において更に下階層の状態遷移を持つと判定されると、その下階層について上記ループを繰り返し実行する。以降同様にして、最下位の状態遷移図に至るまで上記と同様の処理が行われる。このような処理が行われた

結果、未調査の状態が無くなれば、このサブルーチンから復帰し（ステップS42）、HDL生成処理を終了する。

【0054】本実施形態におけるHDL生成処理では、以上のようにして、階層構造を有する拡張状態遷移図における各状態が順次調べられ、設計対象のLSIに対するレジスタ転送レベルのハードウェア記述データが生成される。図29は、このようなHDL生成処理により図28に示した拡張状態遷移図データから得られるハードウェア記述データの一部を示す図である。この図におけるHDLの記述402は、ステップS58で生成される部分であって、下階層の状態遷移を持つマクロ状態HS0で動作していることを示すフラグCLK3MおよびCLK3MMと、マクロ状態HS0の下階層の状態遷移用として供給すべき元のクロックCLKとの論理積をとることにより、クロックのゲーティングを行ってクロックCLK3を生成することを示している。このクロックCLK3が実際に下階層の状態遷移用クロックとして供給される。また、図29におけるHDLの記述403もステップS58で生成され、これは上記フラグCLK3MおよびCLK3MMの作成を示している。

【0055】本実施形態によれば、ステップ58における上記のような処理により、各マクロ状態に対し、設計対象のLSIがそのマクロ状態であるとき以外には、そのマクロ状態の下階層に対して状態遷移のクロックが供給されないような構成を記述したハードウェア記述データが生成される。したがって、このハードウェア記述データに基づいて製造されるLSIでは、各マクロ状態について、そのマクロ状態の下階層の状態遷移が生じない期間中すなわちそのマクロ状態以外の状態となる期間中は、その下階層への状態遷移クロックの供給が停止し、これにより消費電力が低減される。

【0056】上記実施形態では、下階層の状態遷移のクロックをゲーティングすることにより消費電力の低減を図っているが、下階層のレジスタ動作に対して供給されるクロックをゲーティングすることによっても消費電力を低減することができる。すなわち、まず、その下階層の上位の階層の状態遷移図における状態コードがこの状態（その下階層の状態遷移を持つマクロ状態）であることを示す値のときに「1」となり、それ以外のときに「0」となるようなフラグ信号のハードウェア記述データを生成し、次に、このフラグ信号と下階層のレジスタ動作に対して供給すべき元のクロックとの論理積の信号を下階層のレジスタ動作に対するクロックとして供給することを示すハードウェア記述データを生成するようにしてもよい。このようなハードウェア記述データに基づいて製造されるLSIでは、レジスタ動作用として供給すべきクロックが上記フラグを用いてゲーティングされるため、各マクロ状態について、そのマクロ状態以外の状態となる期間中は、その下階層の状態に割り付けられ

たレジスタ動作に対するクロックの供給が停止する。これにより、LSIの消費電力が低減される。

【0057】<変形例>以上において説明した各実施形態は、それぞれ、論理合成の効率とデバッグの容易性などを向上させるために有限状態機械ブロックとデータパス・ブロックを別の階層ブロックにするという特徴や、クロックスキューの発生防止等のために状態レジスタを複数ビット分のフリップフロップから成る一つの回路ブロックとして実現するという特徴、消費電力を低減するために下階層の状態遷移用のクロックのゲーティングを行うという特徴などを有しているが、これらの特徴を複数備えたLSI機能設計支援装置や、これらの特徴を全て備えたLSI機能設計支援装置も実現可能である。

#### 【図面の簡単な説明】

【図1】 本発明の実施形態であるLSI機能設計支援装置のハードウェア構成を示す概略ブロック図。

【図2】 上記LSI機能設計支援装置における処理を示すフローチャート。

【図3】 実施形態1のLSI機能設計支援装置によるHDL生成処理を示すフローチャート。

【図4】 実施形態1のLSI機能設計支援装置に入力される拡張状態遷移図データの一例を示す図。

【図5】 上記拡張状態遷移図データのデータ構造を示す図。

【図6】 従来のLSI機能設計支援装置により上記拡張状態遷移図データから生成されるハードウェア記述データを示す図。

【図7】 図6のハードウェア記述データに対応する設計対象のLSIの外部ポートの入出力関係を示す図。

【図8】 実施形態1において上記拡張状態遷移図データから生成されるハードウェア記述データのうち有限状態機械ブロックの部分を示す図。

【図9】 実施形態1において上記拡張状態遷移図データから生成されるハードウェア記述データのうちデータパス・ブロックの部分を示す図。

【図10】 実施形態1において生成されるハードウェア記述データに対応するLSIの構成を示すブロック図。

【図11】 実施形態2のLSI機能設計支援装置によるHDL生成処理におけるデータパス・ブロックのハードウェア記述の生成を示すフローチャート。

【図12】 実施形態2において上記拡張状態遷移図データから生成されるハードウェア記述データのうち組合せ回路ブロックの部分を示す図。

【図13】 実施形態2において上記拡張状態遷移図データから生成されるハードウェア記述データのうちレジスタ・ブロックの部分を示す図。

【図14】 実施形態2において生成されるハードウェア記述データに対応するLSIの構成のうちデータパス・ブロックの部分を示すブロック図。

【図15】 実施形態3のLSI機能設計支援装置によるHDL生成処理を示すフローチャート。

【図16】 実施形態3において生成されるハードウェア記述データに対応するLSIの構成を示すブロック図。

【図17】 実施形態4のLSI機能設計支援装置によるHDL生成処理を示すフローチャート。

【図18】 実施形態4のLSI機能設計支援装置に入力される状態遷移図データの一例を示す図。

【図19】 実施形態4における上記状態遷移図データの状態コード解析の結果を示す図。

【図20】 実施形態4におけるマクロ・ライブラリに登録されているフリップフロップ・マクロを示す図。

【図21】 実施形態4におけるハードウェア記述データ生成の際に使用されるフリップフロップ・データを示す図。

【図22】 実施形態4のLSI機能設計支援装置に入力される状態遷移図データに対応する、状態コード割り付け後の状態遷移図を示す図。

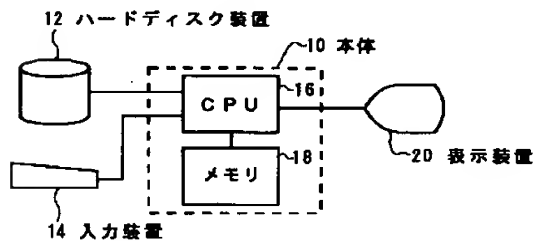
【図23】 従来のLSI機能設計支援装置により上記状態遷移図データから生成されるハードウェア記述データを示す図。

【図24】 実施形態4において上記状態遷移図データから生成されるハードウェア記述データを示す図。

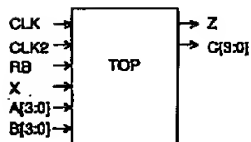
【図25】 従来のLSI機能設計支援装置により生成される図23のハードウェア記述データから論理合成により得られる論理回路を示す図。

【図26】 実施形態4において生成される図24のハードウェア記述データから論理合成により得られる論理 \*

【図1】



【図7】



【図20】

マクロ名	ビット数
R42	4
R82	8

\* 回路を示す図。

【図27】 実施形態5のLSI機能設計支援装置によるHDL生成処理を示すフローチャート。

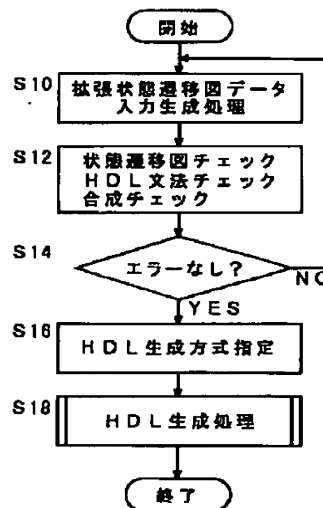
【図28】 実施形態5のLSI機能設計支援装置に入力される拡張状態遷移図データの一例を示す図。

【図29】 実施形態5において図28の拡張状態遷移図データから生成されるハードウェア記述データの一例を示す図。

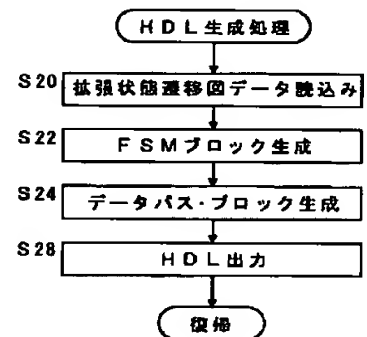
【符号の説明】

- 10 … LSI機能設計支援装置のハードウェア本体  
 12 … ハードディスク装置  
 16 … CPU  
 18 … メモリ  
 211 … 状態コードの割付を示すハードウェア記述 (実施形態4)  
 212 … 状態レジスタを示すハードウェア記述 (実施形態4)  
 213 … 4ビットのフリップフロップ・マクロの使用を示すハードウェア記述 (実施形態4)  
 402 … 状態遷移用クロックのゲーティングを示すハードウェア記述 (実施形態5)  
 403 … 上記ゲーティングに使用されるフラグの作成を示すハードウェア記述 (実施形態5)  
 FSM … 有限状態機械ブロック  
 REGACT … データバス・ブロック  
 COMB … 組合せ回路ブロック  
 REG … レジスタ・ブロック  
 R42 … 4ビット・フリップフロップ・マクロ

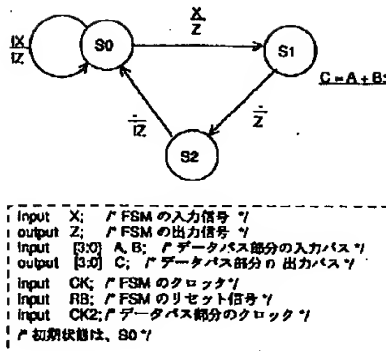
【図2】



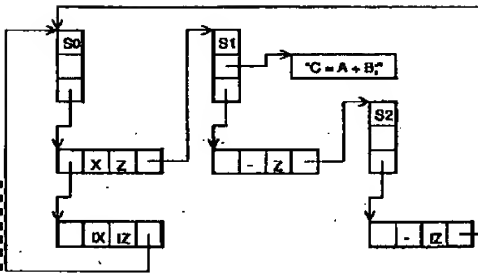
【図3】



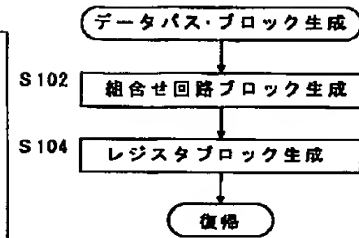
【図4】



【図5】



【図11】



【図8】

【図6】

```

module TOP (CLK, RB, CK2, X, A, B, C, Z);
  input CLK, RB;
  input CK2;
  input [3:0] X;
  input [3:0] A, B;
  output [3:0] C;
  reg [3:0] C;
  output Z;
  reg Z;

  parameter [1:0]
    S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

  reg [1:0] curr_TOP, next_TOP;

  always @(posedge CK2) begin
    case (curr_TOP)
      S0_TOP : begin
        nop;
      end
      S1_TOP : begin
        C = A + B;
      end
      S2_TOP : begin
        nop;
      end
    endcase
  end

  task nop;
  begin
  end
endtask /* nop */

  always @(posedge CLK or negedge RB) begin
    if (!RB) begin
      curr_TOP <= #1 S0_TOP;
    end else begin
      curr_TOP <= #1 next_TOP;
    end

    always @(Z or CK2 or X or A or B
      or C or curr_TOP) begin
      case (curr_TOP)
        S0_TOP : begin
          if (X) begin
            Z = 1;
            next_TOP = S1_TOP;
          end else begin
            Z = 0;
            next_TOP = S0_TOP;
          end
        end
        S1_TOP : begin
          Z = 1;
          next_TOP = S2_TOP;
        end
        S2_TOP : begin
          Z = 0;
          next_TOP = S0_TOP;
        end
      endcase
    end
  end
endmodule

```

```

module FSM (CLK, RB, X, Z, curr_TOP);
  input CLK, RB;
  input X;
  output Z;
  reg Z;
  output [1:0] curr_TOP;
  reg [1:0] curr_TOP, next_TOP;

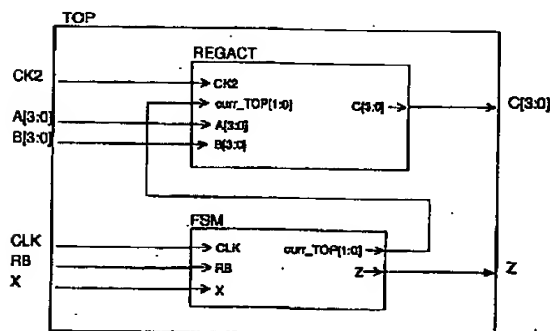
  parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

  always @(posedge CLK or negedge RB) begin
    if (!RB) begin
      curr_TOP <= #1 S0_TOP;
    end else begin
      curr_TOP <= #1 next_TOP;
    end
  end

  always @(Z or X or curr_TOP) begin
    case (curr_TOP)
      S0_TOP : begin
        if (X) begin
          Z = 1;
          next_TOP = S1_TOP;
        end else begin
          Z = 0;
          next_TOP = S0_TOP;
        end
      end
      S1_TOP : begin
        Z = 1;
        next_TOP = S2_TOP;
      end
      S2_TOP : begin
        Z = 0;
        next_TOP = S0_TOP;
      end
    endcase
  end
endmodule

```

【図10】



【図13】

```

module REG (CK2, next_C, C);
  input CK2;
  input [3:0] next_C;
  output [3:0] C;
  reg [3:0] C;

  always @(posedge CK2) begin
    C = next_C;
  end
endmodule

```

【図18】

状態名	次の状態	出力
S0	S1	
S1	S2	Z
S2	S3	Z
S3	S4	
S4	S5	
S5	S6	
S6	S7	Z
S7	S8	Z
S8	S0	

【図9】

```

module REGACT ( CK2, curr_TOP, A, B, C);
input  [1:0]  curr_TOP;
input  [3:0]  A, B;
output [3:0]  C;
reg  [3:0]  C;

parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

always @(posedge CK2) begin
  case (curr_TOP)
    S0_TOP: begin
      nop;
    end
    S1_TOP: begin
      C = A + B;
    end
    S2_TOP: begin
      nop;
    end
  endcase
end

task nop;
begin end
endtask /* nop */

endmodule

```

【図12】

```

module COMB (curr_TOP, A, B, C);
input  [1:0]  curr_TOP;
input  [3:0]  A, B;
output [3:0]  C;
reg  [3:0]  C;

parameter [1:0] S0_TOP = 2'b00, S1_TOP = 2'b01, S2_TOP = 2'b10;

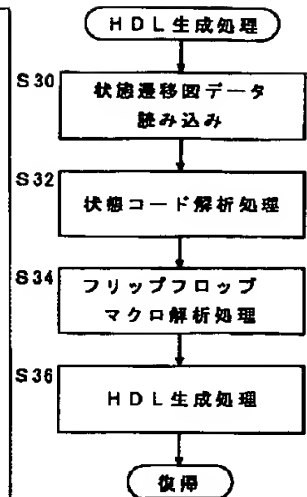
always @(A or B) begin
  case (curr_TOP)
    S0_TOP: begin
      nop;
    end
    S1_TOP: begin
      C = A + B;
    end
    S2_TOP: begin
      nop;
    end
  endcase
end

task nop;
begin end
endtask /* nop */

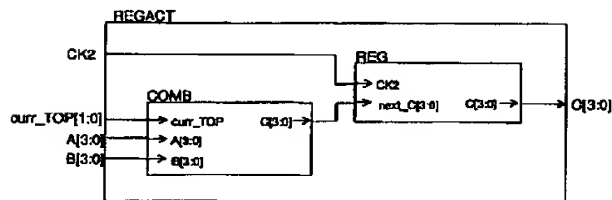
endmodule

```

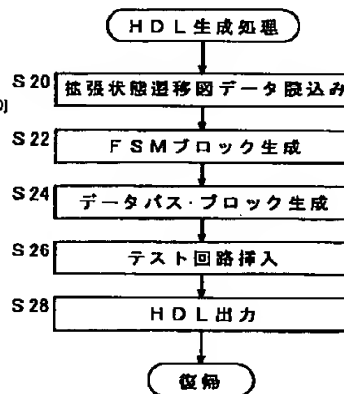
【図17】



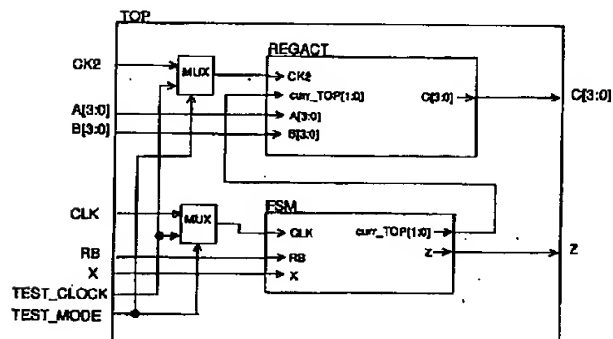
【図14】



【図15】



【図16】



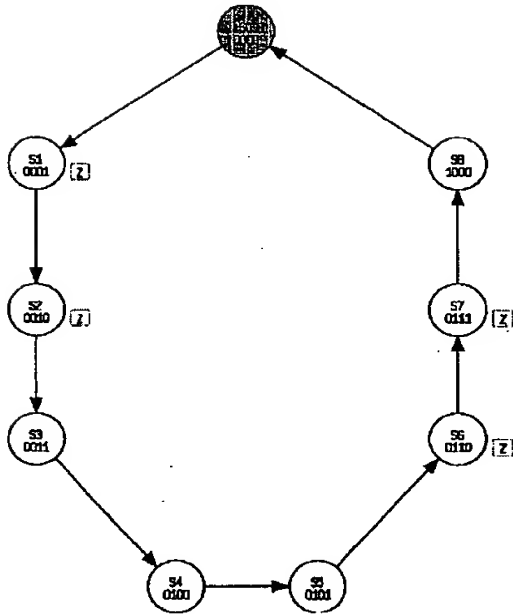
【図19】

(a)		(b)	
状態数	ビット数	状態名	状態コード
9	4	S0	0000
		S1	0001
		S2	0010
		S3	0011
		S4	0100
		S5	0101
		S6	0110
		S7	0111
		S8	1000

【図21】

マクロ名	ビット数
R42	4

【図22】



【図23】

```

module SAMPLE ( CLK, RB, Z);
input  CLK, RB;
output Z;
reg    Z;

parameter [3:0] // synopsys enum state_State
S0_State = 4'b0000,
S1_State = 4'b0001,
S2_State = 4'b0010,
S3_State = 4'b0011,
S4_State = 4'b0100,
S5_State = 4'b0101,
S6_State = 4'b0110,
S7_State = 4'b0111,
S8_State = 4'b1000,
S9_State = 4'b1001;

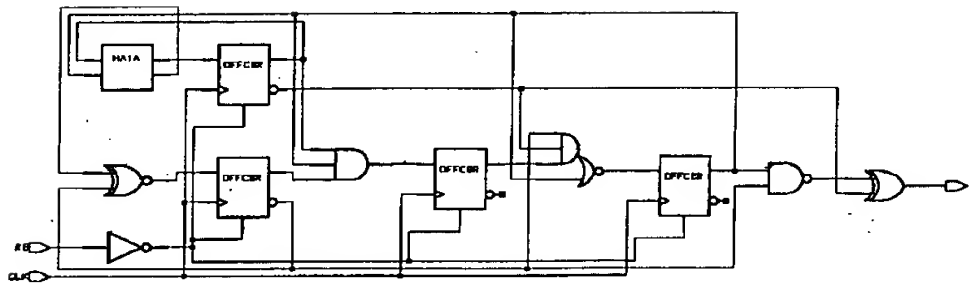
reg [3:0] /* synopsys enum state_State */ State, next_State; // synopsys state_vector State

always @ ( posedge CLK or negedge RB) begin
    if ( !RB) begin
        State <= #1 S0_State;
    end else begin
        State <= #1 next_State;
    end
end

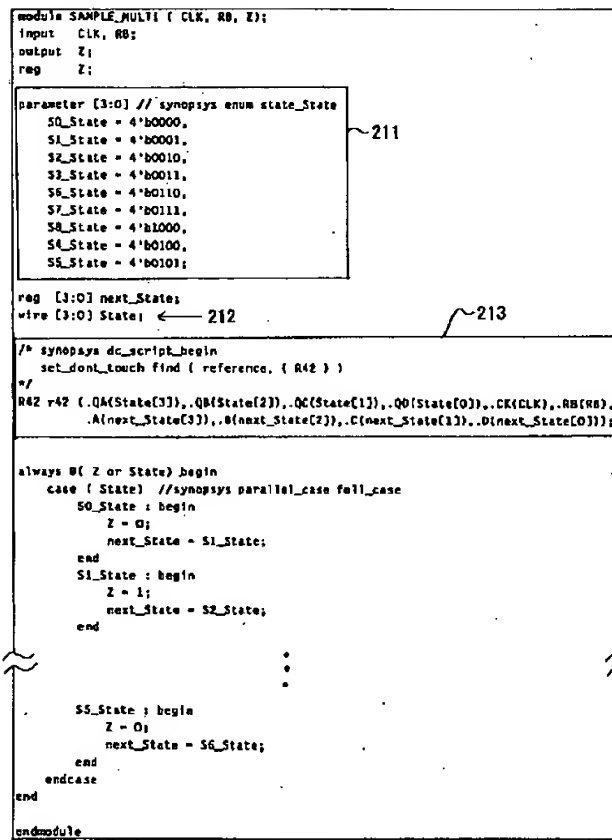
always @ ( Z or State) begin
    case ( State) //synopsys parallel_case full_case
        S0_State : begin
            Z = 0;
            next_State = S1_State;
        end
        S1_State : begin
            Z = 1;
            next_State = S2_State;
        end
        .
        .
        .
        S5_State : begin
            Z = 0;
            next_State = S6_State;
        end
    endcase
end
endmodule

```

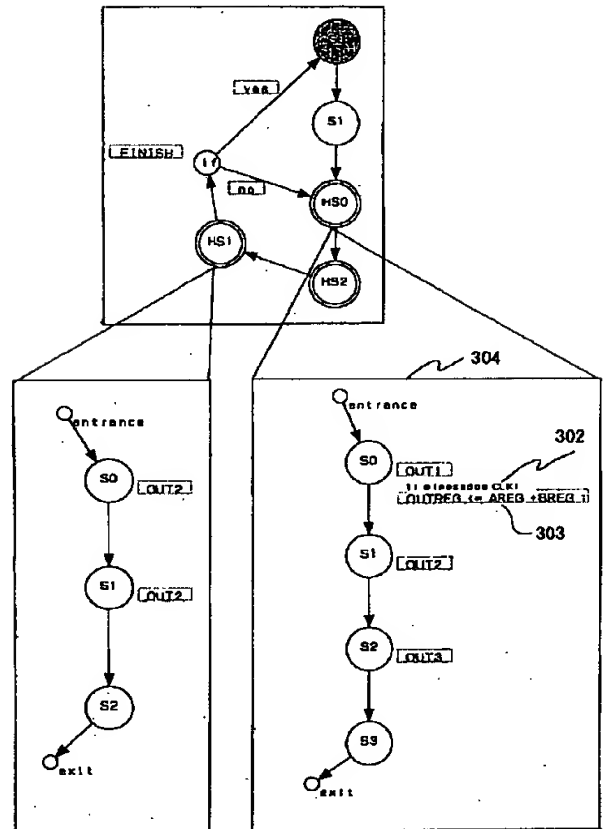
【図25】



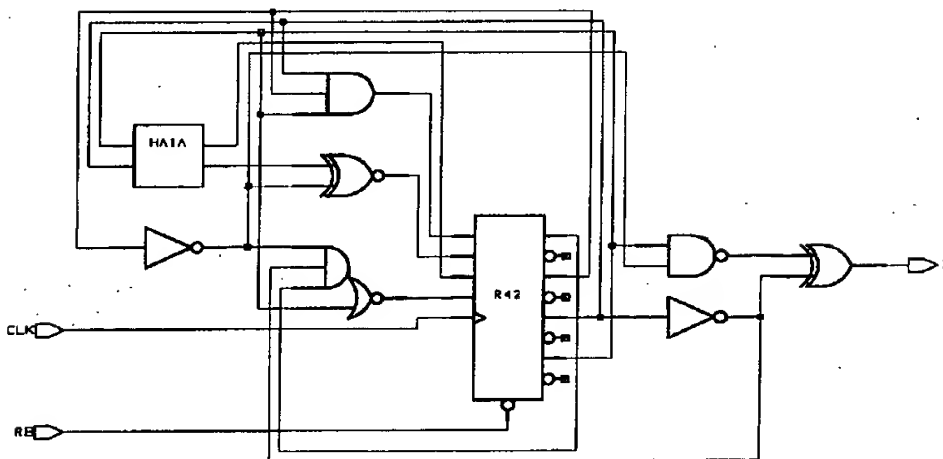
【図24】



【図28】

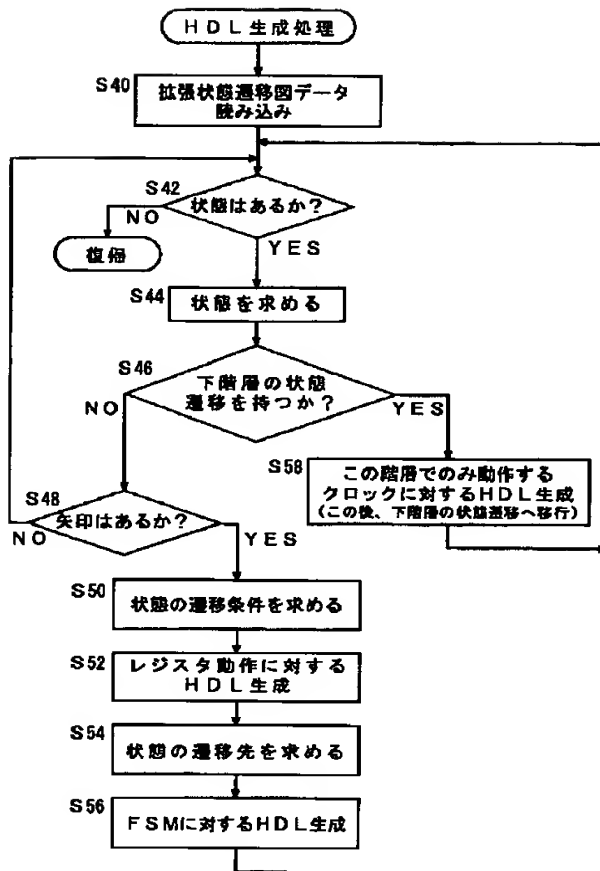


【図26】





【図27】



【図29】

```

module TESTLP ( RB, FINISH, CLK, AREG, BREG, OUTREG, OUT1, OUT2, OUT3);
input RB;
input FINISH, CLK;
input [7:0] AREG,BREG;
output [7:0] OUTREG;
reg [7:0] OUTREG;

output OUT1, OUT2, OUT3;
reg OUT3;
reg OUT1_H50, OUT1_H52, OUT2_H50, OUT2_H52, OUT2_H51;
reg ins_H50_test, fas_H52_test, ins_H51_test, ex_H50, ex_H52, ex_H51;

...

wire CLK3;
assign CLK3 = CLK & (CLK3M[CLK3MM]); ~402

wire CLK4;
assign CLK4 = CLK & (CLK4M[CLK4MM]);

wire CLK5;
assign CLK5 = CLK & (CLK5M[CLK5MM]);

always # ( negedge CLK or negedge RB ) begin
  if ( !RB ) begin
    CLK3M <= 0;
  end else begin
    CLK3M <= !ex_H50 | ins_H50_test;
  end
end

always # ( negedge CLK or negedge RB ) begin
  if ( !RB ) begin
    CLK3MM <= 0;
  end else begin
    CLK3MM <= CLK3M;
  end
end

always # ( posedge CLK3 or negedge RB ) begin
  if ( !RB ) begin
    curr_H50 <= #1 idle_H50;
  end else begin
    curr_H50 <= #1 next_H50;
  end
end
end
  
```